



Tower of covering arrays

Jose Torres-Jimenez^{a,b,*}, Idelfonso Izquierdo-Marquez^a, Raghu N. Kacker^b,
D. Richard Kuhn^b

^a CINESTAV-Tamaulipas, Information Technology Laboratory, Km. 5.5 Carretera Cd. Victoria-Soto la Marina, 87130, Cd. Victoria Tamps., Mexico

^b National Institute of Standards and Technology Gaithersburg, MD 20899-8910, USA

ARTICLE INFO

Article history:

Received 12 August 2013

Received in revised form 2 March 2015

Accepted 25 March 2015

Available online xxxx

Keywords:

Covering arrays

Tower of covering arrays

ABSTRACT

Covering arrays are combinatorial objects that have several practical applications, specially in the design of experiments for software and hardware testing. A covering array of strength t and order v is an $N \times k$ array over $\mathbb{Z}_v$ with the property that every $N \times t$ subarray covers all members of $\mathbb{Z}_v^t$ at least once. In this work we explore the construction of a Tower of Covering Arrays (TCA) as a way to produce covering arrays that improve or match some current upper bounds. A TCA of height h is a succession of $h + 1$ covering arrays $C_0, C_1, \dots, C_h$ in which for $i = 1, 2, \dots, h$ the covering array C_i is one unit greater in the number of factors and the strength of the covering array C_{i-1} ; this way, if the covering array C_0 is of strength t and has k factors then the covering arrays $C_1, \dots, C_h$ are of strength $t + 1, \dots, t + h$ and have $k + 1, \dots, k + h$ factors respectively. We note that the ratio between the number of rows of the last covering array C_h in a TCA of height h and the number of rows of the best known covering array for the same values of t, k , and v as for C_h is reduced as h grows. Therefore, we search for TCAs with the greatest height possible. The relevant results are the improvement of nineteen current upper bounds for $v = 2$ and $t \in \{7, 8, 9, 10, 11\}$, and the construction of twenty-one covering arrays that matched current upper bounds.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

Let N, t, k , and v be four positive integers, a covering array $CA(N; t, k, v)$ is an $N \times k$ matrix over $\mathbb{Z}_v = \{0, 1, \dots, v - 1\}$ with the property that every $N \times t$ subarray covers at least once each t -tuple of the set $\mathbb{Z}_v^t$. The k columns of a covering array are called *factors* or *components*, v is the *order* or the number of symbols for each component, and t is the *strength* of the covering array. In a covering array of strength t and order v , every subset of t factors get combined in all the v^t ways at least once. This is very useful for the design of experiments because it ensures that all possible interactions among any t factors are checked in the experiment. For this reason covering arrays have been used in software and hardware testing [4,8,7].

The covering array construction problem consists of generating a covering array $CA(N; t, k, v)$ given the parameters t, k , and v in such a way that the number of rows N of the covering array is the minimum possible. The smallest N for which a covering array exists is the covering array number for the parameters t, k , and v , and it is denoted by $CAN(t, k, v) = \min\{N : \exists CA(N; t, k, v)\}$. In this work we propose a method to generate covering arrays based on the construction of a *Tower of Covering Arrays (TCA)*.

* Corresponding author at: CINESTAV-Tamaulipas, Information Technology Laboratory, Km. 5.5 Carretera Cd. Victoria-Soto la Marina, 87130, Cd. Victoria Tamps., Mexico. Tel.: +52 8341070220; fax: +52 8343147392.

E-mail addresses: jtj@cinvestav.mx (J. Torres-Jimenez), iizquierdo@tamps.cinvestav.mx (I. Izquierdo-Marquez), raghu.kacker@nist.gov (R.N. Kacker), kuhn@nist.gov (D. Richard Kuhn).

<http://dx.doi.org/10.1016/j.dam.2015.03.010>

0166-218X/© 2015 Elsevier B.V. All rights reserved.

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \quad \delta = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad X = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \quad \left(\begin{array}{ccc|c} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{array} \right)$$

Fig. 1. The construction $\mathcal{T}$ applied to the base CA(4; 2, 3, 2) and to the matrix δ shown in the figure, produces the covering array of strength three CA(8; 3, 4, 2).

We define a TCA of height h as a sequence of $h + 1$ covering arrays $C_0, C_1, \dots, C_h$, where C_0 is a covering array of strength t and k factors called the *base* of the TCA, and for $i = 1, 2, \dots, h$, C_i is a covering array of strength $t + i$ and $k + i$ factors. The construction of a TCA relates in some sense to: the *extension of coverage* of a testing array [5], the *embedding problem* of a covering array [5], and the *Zero-Sum* construction [1,6]. The main difference with the TCA construction is that the strength and the number of factors of an existing CA is increased in one unit, meanwhile the *extension of coverage* adds coverage to an existing array without increasing the strength and/or the number of factors; the *embedding problem* does not modify either the strength or the number of factors of an array; and the *Zero-Sum* construction only increases the number of factors but not the strength of a CA.

Let A be a CA($N; t, k, v$), and let A_j ($0 \leq j \leq k - 1$) be one column of A . To translate the column A_j by a value $c \in \mathbb{Z}_v$ means to add modulo v the value c to every element of the column A_j . The symbol $\oplus$ will be used to denote the operation of column translation.

The covering arrays in a TCA are created by the iterative application of one construction we called $\mathcal{T}$ (defined below). This construction takes a base covering array CA($N; t, k, v$) denoted by A of strength t , N rows, k columns, and order v , and produces a covering array CA($Nv; t + 1, k + 1, v$) of strength $t + 1$, Nv rows, $k + 1$ columns, and order v , by juxtaposing vertically v copies of the base covering array, but translating the j th column of the i th copy by a value $\delta_{i,j} \in \mathbb{Z}_v$ this CA is denoted by B . The column $k + 1$ of the covering array B is formed by N zeros, followed by N ones, and so on until finish with N elements equal to $v - 1$. The construction $\mathcal{T}$ can be applied iteratively to the covering array CA($Nv; t + 1, k + 1, v$) of strength $t + 1$ to try to produce a covering array CA($Nv^2; t + 2, k + 2, v$) of strength $t + 2$. The process would continue expanding the covering array of strength $t + 2$ to a covering array of strength $t + 3$, and so on.

To see how the construction $\mathcal{T}$ can produce a covering array of strength $t + 1$ given a covering array of strength t , consider the CA(4; 2, 3, 2) and the matrix δ ($\delta_{i,j}$ is used to do the translation of the i th copy of the CA) in Fig. 1. The application of the construction $\mathcal{T}$ to this base covering array using this matrix δ produces the covering array CA(8, 3, 4, 2).

The concept of TCAs was introduced in some way for an orthogonal array (OA). In the book of Hedayat, Sloane, and Stufken [6] the theorem 2.24 stated that an OA($N; 2u, k, 2$) (N is the number of rows, $2u$ is the strength, k is the number of factors, and 2 is the order) exists if and only if an OA($2N; 2u + 1, k + 1, 2$) exists. The converse of this theorem is that from an orthogonal array OA($N; 2u, k, 2$) one can construct an orthogonal array OA($2N; 2u + 1, k + 1, 2$). This construction enables the generation of a tower of orthogonal arrays of height $h = 1$, but they only exist for $v = 2$ and when the strength of the base orthogonal array is even ($t = 2u$). In this work we seek TCAs in which the base covering array can have any strength and any order.

The remainder of the document is organized as follows: Section 2 shows how to construct a TCA using the construction $\mathcal{T}$; Section 3 shows the relevant TCAs constructed by first generating all the non-isomorphic base CAs for the parameters N, k, t , and v ; Section 4 shows the TCAs constructed from a given base covering array constructed with a simulated annealing algorithm reported in the literature; and Section 5 presents the conclusions.

2. Construction of a TCA

To begin, we introduce the concept of isomorphism in covering arrays: any of the following three operations produces a covering array *isomorphic* to the covering array over which the operation is applied: (a) To permute the rows of the covering array; (b) To permute the columns of the covering array; and (c) To permute the symbols in the columns of the covering array. Furthermore, any combination of these three operations applied to a covering array produces an isomorphic covering array. For a covering array with N rows, k columns, and order v there are $N! k! (v!)^k$ isomorphic CAs (even some of these covering arrays may be identical).

Given a base covering array CA($N; t, k, v$) denoted by A the first k columns of the resulting matrix B produced by the construction $\mathcal{T}$ are formed by juxtaposing vertically v covering arrays $\overline{A^0}, \overline{A^1}, \dots, \overline{A^{v-1}}$ isomorphic to A , where each covering array $\overline{A^i}$ is derived from A by translating the columns of A by the values at row i of the matrix δ .

The construction $\mathcal{T}$ does not always produce a covering array of strength $t + 1$ based on a covering array of strength t . So, it is not always possible to construct a TCA with height $h > 0$. But when a TCA with $h > 0$ is constructed we noted that the ratio between the number of rows of the last covering array in the TCA and the number of rows of the best known covering array for the same values of t, k and v decreases as h grows. Therefore we are interested in constructing a TCA with the greatest height possible, since it has more chance to produce covering arrays that improve or match a current upper bound.

Table 1
Percentage of verifications omitted for the matrices B .

$CA(N; t, k, v)$	$v^{k(v-1)}$	Verified	Not verified	Percentage of verifications omitted
$CA(6; 2, 5, 2)$	32	18	14	43.75%
$CA(6; 2, 7, 2)$	128	19	109	85.16%
$CA(6; 2, 10, 2)$	1024	23	1001	97.75%
$CA(7; 2, 11, 2)$	2048	31	2017	98.49%
$CA(7; 2, 15, 2)$	32768	45	32723	99.86%
$CA(8; 2, 11, 2)$	2048	30	2018	98.54%
$CA(8; 2, 16, 2)$	65536	119	65417	99.82%
$CA(8; 2, 20, 2)$	1048576	122	1048454	99.99%
$CA(9; 2, 13, 2)$	8192	194	7998	97.63%
$CA(10; 3, 5, 2)$	32	24	8	25.00%
$CA(12; 3, 7, 2)$	128	30	98	76.56%
$CA(12; 3, 11, 2)$	2048	48	2000	97.66%
$CA(15; 3, 12, 2)$	4096	61	4035	98.51%
$CA(24; 4, 10, 2)$	1024	68	956	93.36%
$CA(24; 4, 12, 2)$	4096	81	4015	98.02%
$CA(9; 2, 4, 3)$	6561	4637	1924	29.32%
$CA(11; 2, 5, 3)$	59049	12522	46527	78.79%
$CA(12; 2, 7, 3)$	4782969	115957	4667012	97.58%
$CA(13; 2, 9, 3)$	387420489	1514159	385906330	99.61%
$CA(16; 2, 5, 4)$	1073741824	184673582	889068242	82.80%

The application of the construction $\mathcal{T}$ to all matrices δ for a covering array $CA(N; t, k, v)$ requires to generate and verify $v^{k(v-1)}$ matrices B . The generation of a matrix B is done in a time linear in the size of the matrix, i.e., $O(Nv(k+1))$. The verification whether a matrix B is a covering array of strength $t+1$ requires one to check that every submatrix of $t+1$ columns of B covers all the tuples of the set $\mathbb{Z}_v^{t+1}$. Each submatrix has dimensions $(Nv) \times (t+1)$ and there are $\binom{k+1}{t+1}$ of such submatrices; since the submatrix verification is performed in linear time, the computational cost of checking whether a matrix B is a covering array of strength $t+1$ is of order $O\left(\binom{k+1}{t+1}Nv(t+1)\right)$, then, the cost of applying the construction $\mathcal{T}$ to the $v^{k(v-1)}$

different matrices δ that exist for a base covering array $CA(N; t, k, v)$ is of order $O\left(v^{k(v-1)}\left[Nv(k+1) + \binom{k+1}{t+1}Nv(t+1)\right]\right)$.

To improve the execution time, the approach followed was to represent a matrix δ as a linear vector (also called δ) concatenating the rows of the matrix (excluding the first row since all its elements are zero). In addition, the vector is not generated sequentially, but rather it is generated in (v, k) -Gray code using the algorithm in [3]. For each vector the construction $\mathcal{T}$ is applied and the resulting B matrix is verified to see whether it is a covering array of strength $t+1$. Generating the vector δ in Gray code has the following advantages:

1. The application of the construction $\mathcal{T}$ is accelerated notably, since the matrix B for a vector δ is different from the matrix B for the previous vector δ in only one column.
2. Suppose the current matrix B is not a covering array of strength $t+1$ because the combination of $t+1$ columns $\{l_0, l_1, \dots, l_t\}$, where $0 \leq l_0, l_1, \dots, l_t \leq k-1$, does not cover all tuples of $\mathbb{Z}_v^{t+1}$. Given that, for the following matrices B only one column is updated, we can omit the verification of the following matrices B until we get a matrix B whose updated column j is in the set $\{l_0, l_1, \dots, l_t\}$.

Following this approach the cost of generating the matrices B is $O(Nv \times (k+1))$ for the first vector δ , and $O(N)$ for all the other vectors δ . So, the cost of generating the matrices B for all the vectors δ is $O(Nv(k+1) + N(v^{k(v-1)} - 1))$.

Table 1 shows the number of times the verification process is omitted for a set of 20 covering arrays. The first column of the table is the base covering array, the second column is the number of vectors δ for the base covering array, the third column is the number of matrices B for which the verification process was performed, the fourth column is the number of matrices B for which the verification was omitted, and the last column is the percentage of verifications omitted. In the table we can see that the greater percentage of verifications omitted occurs for the matrices B with a small number of columns, say $k \leq 8$, because there is more chance that the updated column of matrix B is one of the columns of the submatrix $\{l_0, l_1, \dots, l_t\}$ having missing tuples. For greater values of k , say $k \geq 9$, the number of matrices B verified is much smaller than the number of matrices B whose verification was omitted. Therefore the generation of δ in v -ary Gray code allows important time savings when the construction $\mathcal{T}$ is applied.

3. TCAs from non-isomorphic base CAs

The non-isomorphic covering arrays are covering arrays having the same parameters N, t, k , and v that cannot be transformed among them by means of the three operations that produce equivalent covering arrays, i.e., permutations of rows, permutations of columns, and permutation of the symbols in the columns.

The isomorphic covering arrays are equivalent, and all of them produce a TCA with the same maximum height. The non-isomorphic covering arrays can produce a TCA with different maximum height. So, for a given combination of the

Table 2
TCAs from non-isomorphic base CAs.

Base covering array	TCA	Best known CAs
CA(7; 2, 5, 2)	CA(224; 7, 10, 2)	CA(274; 7, 10, 2)
	CA(112; 6, 9, 2)	CA(108; 6, 9, 2)
	CA(56; 5, 8, 2)	CA(52; 5, 8, 2)
	CA(28; 4, 7, 2)	CA(24; 4, 7, 2)
	CA(14; 3, 6, 2)	CA(12; 3, 6, 2)
	CA(7; 2, 5, 2)	CA(6; 2, 5, 2)
CA(8; 2, 11, 2)	CA(256; 7, 16, 2)	CA(≥ 762 ; 7, 16, 2)
	CA(128; 6, 15, 2)	CA(128; 6, 15, 2)
	CA(64; 5, 14, 2)	CA(64; 5, 14, 2)
	CA(32; 4, 13, 2)	CA(32; 4, 13, 2)
	CA(16; 3, 12, 2)	CA(15; 3, 12, 2)
	CA(8; 2, 11, 2)	CA(7; 2, 11, 2)
CA(5; 2, 4, 2)	CA(10; 3, 5, 2)	CA(10; 3, 5, 2)
	CA(5; 2, 4, 2)	CA(5; 2, 4, 2)
CA(6; 2, 5, 2)	CA(12; 3, 6, 2)	CA(12; 3, 6, 2)
	CA(6; 2, 5, 2)	CA(6; 2, 5, 2)
CA(12; 2, 11, 2)	CA(24; 4, 12, 2)	CA(24; 4, 12, 2)
	CA(12; 3, 11, 2)	CA(12; 3, 11, 2)
CA(21; 4, 6, 2)	CA(42; 5, 7, 2)	CA(42; 5, 7, 2)
	CA(21; 4, 6, 2)	CA(21; 4, 6, 2)
CA(11; 2, 5, 3)	CA(33; 3, 6, 3)	CA(33; 3, 6, 3)
	CA(11; 2, 5, 3)	CA(11; 2, 5, 3)
	CA(351; 5, 7, 3)	CA(351; 5, 7, 3)
CA(13; 2, 4, 3)	CA(117; 4, 6, 3)	CA(111; 4, 6, 3)
	CA(39; 3, 5, 3)	CA(33; 3, 5, 3)
	CA(13; 2, 4, 3)	CA(9; 2, 4, 3)
CA(15; 2, 5, 3)	CA(405; 5, 8, 3)	CA(405; 5, 8, 3)
	CA(135; 4, 7, 3)	CA(123; 4, 7, 3)
	CA(45; 3, 6, 3)	CA(33; 3, 6, 3)
	CA(15; 2, 5, 3)	CA(11; 2, 5, 3)

Table 3
Upper bounds improved by means of the construction of TCAs from non-isomorphic base CAs.

t	k	v	Previous N	New N
7	10	2	274	224
7	11	2	386	256
7	12	2	506	256
7	13	2	634	256
7	14	2	762	256
7	15	2	≥ 762	256
7	16	2	≥ 762	256

parameters N , t , k , and v we use as base CA of the TCA all the non-isomorphic covering arrays. A TCA is constructed for each non-isomorphic base, and the TCA of maximum height for N , t , k , and v is the TCA of greatest height among all TCAs that one have constructed.

To generate the non-isomorphic base CAs we used the algorithm described in [9]. Basically, this algorithm constructs the non-isomorphic covering arrays column by column from 1 column to k columns. The columns are generated in lexicographical order and only the subarrays of minimum lexicographical order are extended. The lexicographical order of a covering array $CA(N; t, k, v)$ is the sequence obtained concatenating the elements of the CA row by row.

The first two TCAs in Table 2 enables the improvement of seven current upper bounds. Table 3 shows these upper bounds, the previous upper bounds were taken from [2]. The exact value of the current upper bound for $\{t = 7, k = 15, v = 2\}$ and $\{t = 7, k = 16, v = 2\}$ are not given explicitly in [2], but they must be greater than or equal to 762 the current upper bound for $\{t = 7, k = 14, v = 2\}$. This follows from the inequality $CAN(t, k, v) \geq CAN(t, k - 1, v)$.

The other TCAs in Table 2 do not improve an upper bound, but one or more of the covering arrays in those TCAs match an upper bound. Moreover, the first two TCAs that improve an upper bound also have some covering arrays that match an upper bound. Table 4 shows the upper bounds matched by means of the construction of TCAs. The first column of Table 4 is the base covering array of the TCA that allows to match the upper bound in the third column of the same table; the second column of the table is the height at which the TCA matched the upper bound in the third column.

Table 4

Upper bounds matched by the TCAs constructed from non-isomorphic base CAs.

Base covering array	Height	Upper bound matched
CA(8; 2, 8, 2)	3	CA(64; 5, 11, 2)
CA(8; 2, 8, 2)	4	CA(128; 6, 12, 2)
CA(8; 2, 9, 2)	3	CA(64; 5, 12, 2)
CA(8; 2, 9, 2)	4	CA(128; 6, 13, 2)
CA(8; 2, 10, 2)	3	CA(64; 5, 13, 2)
CA(8; 2, 10, 2)	4	CA(128; 6, 14, 2)
CA(8; 2, 11, 2)	2	CA(32; 4, 13, 2)
CA(8; 2, 11, 2)	3	CA(64; 5, 14, 2)
CA(8; 2, 11, 2)	4	CA(128; 6, 15, 2)
CA(5; 2, 4, 2)	0	CA(5; 2, 4, 2)
CA(5; 2, 4, 2)	1	CA(10; 3, 5, 2)
CA(6; 2, 5, 2)	0	CA(6; 2, 5, 2)
CA(6; 2, 5, 2)	1	CA(12; 3, 6, 2)
CA(12; 3, 11, 2)	0	CA(12; 3, 11, 2)
CA(12; 3, 11, 2)	1	CA(24; 4, 12, 2)
CA(21; 4, 6, 2)	0	CA(21; 4, 6, 2)
CA(21; 4, 6, 2)	1	CA(42; 5, 7, 2)
CA(11; 2, 5, 3)	0	CA(11; 2, 5, 3)
CA(11; 2, 5, 3)	1	CA(33; 3, 6, 3)
CA(13; 2, 4, 3)	3	CA(351; 5, 7, 3)
CA(15; 2, 5, 3)	3	CA(405; 5, 8, 3)

4. TCAs from a given covering array

The TCAs can also be constructed from a given base covering array. The generation of all non-isomorphic base CAs takes reasonable time only for small values of N , t , k , and v ; so in order to construct TCAs from larger base covering arrays we can construct the base covering arrays with another method. The disadvantage of this approach is that we do not test all the non-isomorphic CAs for the given parameters values. We use the algorithm of simulated annealing developed in [10] to generate the base covering arrays of the TCAs. The relevant results are as follows: five TCAs that improved twelve current upper bounds for $v = 2$ and $t \in \{8, 9, 10, 11\}$. Table 5 shows the TCAs constructed; and Table 6 summarizes the upper bounds improved with these TCAs. The values in the fourth column of Table 6 were taken from [2].

5. Conclusions

In this work we presented the construction of a tower of covering arrays (TCA) to generate covering arrays competitive with the best known ones. We defined a TCA of height h as a sequence of $h + 1$ covering arrays $C_0, C_1, \dots, C_h$, where the covering array C_0 is the base CA of the TCA and for $i \in 1, 2, \dots, h$ the covering array C_i has one more unit of strength than the covering array C_{i-1} . We introduced a construction called $\mathcal{T}$ to construct the TCAs; this construction takes a base covering array $CA(N; t, kv)$ of strength t and sometimes can generate a covering array $CA(Nv; t + 1, k + 1, v)$ of strength $t + 1$.

In order to construct a TCA of the maximum height possible, the construction $\mathcal{T}$ can be applied to every matrix δ that exists for the given base covering array $CA(N; t, k, v)$. The computational cost of this operation is $O(v^{k(v-1)}[Nv(k+1) + \binom{k+1}{t+1}Nv(t+1)])$, but this cost can be reduced if (1) a matrix δ is transformed to a vector (also called δ) in which the rows of the matrix are concatenated in order (excluding the first row given that it is all zeros), and (2) the vector δ is generated in v -ary Gray code. Following this approach the cost of generating the matrices B is $O(Nv(k+1) + N(v^{k(v-1)} - 1))$, and the cost of verify the matrices B is reduced significantly because a great number of them are not verified. We have found that as the number of columns of the base covering array grows, the percentage of the matrices B whose verification is omitted also grows.

To construct a TCA we followed two approaches: for small values of N , t , k , and v , we generate all the non-isomorphic covering arrays $CA(N; t, k, v)$ with the objective of using these covering arrays as the base CA of a TCA; the relevant TCAs obtained enables us to improve seven current upper bounds and to match twenty-one known upper bounds. For larger values of N , t , k , and v , we used a previously reported simulated annealing algorithm to construct the base CAs of the TCAs; in this case the TCAs constructed allow the improvement of twelve upper bounds.

Acknowledgments

The authors acknowledge the GENERAL COORDINATION OF INFORMATION AND COMMUNICATIONS TECHNOLOGIES (CGSTIC) at CINVESTAV for providing HPC resources on the Hybrid Cluster Supercomputer "Xiuhoatl", that have contributed to the research results reported. The following projects have funded the research reported in this paper: 51623—Fondo Mixto CONACyT y Gobierno del Estado de Tamaulipas; 238469—CONACyT Métodos Exactos para Construir Covering Arrays Óptimos; 232987—CONACyT Conjuntos de Prueba Óptimos para Métodos Combinatorios (Optimal Test Sets for Combinatorial Methods). This paper was completed during the first author's stay as a guest researcher at the US National Institute of Standards and Technology (NIST), Gaithersburg MD 20899 USA.

Table 5
TCAs from a given base covering array.

Base covering array	TCA	Best known CAs
CA(27; 4, 7, 2)	CA(108; 6, 9, 2)	CA(108; 6, 9, 2)
	CA(54; 5, 8, 2)	CA(52; 5, 8, 2)
	CA(27; 4, 7, 2)	CA(24; 4, 7, 2)
CA(85; 6, 8, 2)	CA(170; 7, 9, 2)	CA(170; 7, 9, 2)
	CA(85; 6, 8, 2)	CA(85; 6, 8, 2)
CA(112; 6, 9, 2)	CA(3584; 11, 14, 2)	CA(5190; 11, 14, 2)
	CA(1792; 10, 13, 2)	CA(2491; 10, 13, 2)
	CA(896; 9, 12, 2)	CA(1230; 9, 12, 2)
	CA(448; 8, 11, 2)	CA(563; 8, 11, 2)
	CA(224; 7, 10, 2)	CA(274; 7, 10, 2)
	CA(112; 6, 9, 2)	CA(108; 6, 9, 2)
	CA(3840; 11, 15, 2)	CA(≥ 5190 ; 11, 15, 2)
CA(120; 6, 10, 2)	CA(1920; 10, 14, 2)	CA(4081; 10, 14, 2)
	CA(960; 9, 13, 2)	CA(2002; 9, 13, 2)
	CA(480; 8, 12, 2)	CA(795; 8, 12, 2)
	CA(240; 7, 11, 2)	CA(386; 7, 11, 2)
	CA(120; 6, 10, 2)	CA(116; 6, 10, 2)
	CA(4096; 11, 16, 2)	CA(≥ 5190 ; 11, 16, 2)
	CA(2048; 10, 15, 2)	CA(≥ 4081 ; 10, 15, 2)
CA(128; 6, 11, 2)	CA(1024; 9, 14, 2)	CA(3014; 9, 14, 2)
	CA(512; 8, 13, 2)	CA(1051; 8, 13, 2)
	CA(256; 7, 12, 2)	CA(506; 7, 12, 2)
	CA(128; 6, 11, 2)	CA(118; 6, 11, 2)

Table 6
Upper bounds improved by means of the construction of TCAs from a given base covering array.

t	k	v	Previous N	New N
8	11	2	563	448
9	12	2	1230	896
10	13	2	2491	1792
11	14	2	5190	3584
8	12	2	795	480
9	13	2	2002	960
10	14	2	4081	1920
11	15	2	≥ 5190	3840
8	13	2	1051	512
9	14	2	3014	1024
10	15	2	≥ 4081	2048
11	16	2	≥ 5190	4096

Disclaimer

Any mention of commercial products in this paper is for information only; it does not imply recommendation or endorsement by National Institute of Standards and Technology (NIST).

References

- [1] C.J. Colbourn, J.H. Dinitz, *Handbook of Combinatorial Designs*, second ed., Chapman and Hall/CRC, 2006.
- [2] C.J. Colbourn, G. Kéri, P.P.R. Soriano, J.C. Schlage-Puchta, Covering and radius-covering arrays: Constructions and classification, *Discrete Appl. Math.* 158 (11) (2010) 1158–1180.
- [3] D. Guan, Generalized gray codes with applications, in: *Proc. Natl. Sci. Conf. ROC(A)*, vol. 22, 1998, pp. 841–848.
- [4] A. Hartman, Software and hardware testing using combinatorial covering suites, in: M.C. Golumbic, I.B.-A. Hartman (Eds.), *Graph Theory, Combinatorics and Algorithms*, in: *Operations Research/Computer Science Interfaces Series*, vol. 34, Springer US, 2005, pp. 237–266.
- [5] A. Hartman, L. Raskin, Problems and algorithms for covering arrays, *Discrete Math.* 284 (1–3) (2004) 149–156.
- [6] A.S. Hedayat, N.J.A. Sloane, J. Stufken, *Orthogonal Arrays: Theory and Applications*, Springer-Verlag, 1999.
- [7] D.R. Kuhn, R.N. Kacker, Y. Lei, *Practical Combinatorial Testing*, Nist special publication 800-142, National Institute of Standards and Technology, 2010.
- [8] D.R. Kuhn, R.N. Kacker, Y. Lei, J. Hunter, Combinatorial software testing, *Computer* 42 (8) (2009) 94–96.
- [9] J. Torres-Jimenez, I. Izquierdo-Marquez, Construction of non-isomorphic covering arrays, *J. Combin. Theory Ser. A* (2014) submitted for publication.
- [10] J. Torres-Jimenez, E. Rodriguez-Tello, New bounds for binary covering arrays using simulated annealing, *Inform. Sci.* 185 (1) (2012) 137–152.